

REMARKS

This Amendment Under 37 C.F.R. §1.116 is in response to the final Office Action mailed March 29, 2006. The present amendment is filed together with the proper fee for an extension of time within third month and is filed concurrently with a Request for Continued Examination (RCE).

In the final Office Action, claims 1-25, 71-79, 81-90 and 94-97 were rejected are rejected under 35 U.S.C. §102(e) as being anticipated by Rabin et al. (U.S. Patent No. 6,697,948 B1). Claims 1-79 and 81-97 are currently pending.

Independent claim 1

Rabin et al. identifies “an instance of a software as follows, at Col. 3, lines 29-36:

Using this invention, a software vendor may have a specific piece of software, such as a specific application program or a specific book or song, which the vendor wishes to sell or lease, or otherwise distribute in a controlled manner, to users. Each particular copy of the software which is intended to be installed on or used on a user's device, is referred to as an instance of that software, or as a software instance.

and, in Col. 10, lines 31-67, Rabin et al. tell us that there is one unique tag for each piece of software.

Embodiments of the invention also encompass a tag server that accepts a copy of specific vendor software and produces a plurality of tags, one tag per instance of the software, with each tag uniquely identifying an instance of software with which it is associated. Each tag preferably comprises at least one of the name of the software associated with the tag, a unique number of the instance of software associated with the tag, and hash function values computed on portions of the instance of software associated with the tag. A digital signature mechanism may be used to digitally sign the tags and to securely transmit the tags to an intended receiver, such as a user device or guardian center or to the software vendor.

Methods encompassed by the invention include a method for supervising usage of software. The method includes the steps of creating an instance of software and creating a tag that is uniquely associated with the instance of software. The method then distributes the instance of software and securely distributes the tag to a user device and receives the instance of software and the associated tag at the user device. The method then detects an attempt to use the instance of the software on the user device and

determines if the attempt to use the instance of the software is allowable by determining a status of the tag that is associated with the instance of software to be used.

In the method, tag creation includes steps of assigning a unique number to the instance of software and computing a first hash function value on portions of the content of the instance of software. Then computing a second hash function value for the instance of software, the second hash function value combining the name of the software, the unique number of the instance of software, and the first hash function value. Next, the method includes the step of computing a tag that is uniquely associated with the instance of software, the tag including the name of the software, the unique number of the instance of software and the second hash value.

In Rabin et al., the same piece of software located on different user machines will not receive the same tag. In fact, Rabin et al.'s tags are unique to the piece of software, to the instance of software, and is unique for a same piece of software across users. That is, Rabin et al. teach that different copies of the same piece of software installed on different user devices (what Rabin et al. call an "instance of software") are assigned different tags. In Rabin et al., therefore, each tag for each instance of software is unique – meaning that each different piece of software installed on different user devices has its own unique tag.

In contrast, claim 1 has been amended to recite that **"identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are associated with identical identifiers and are code signed with identical PKI certificates."**

Claim 1, therefore, distinguishes over the applied reference to Rabin et al. Reconsideration and withdrawal of the rejections applied to claim 1 and its dependent claims are, therefore, respectfully requested.

Independent claim 17

Claim 17 has been amended in manner that is similar to the manner in which independent claim 1 was amended:

producing a separate and unique PKI certificate for each of the plurality of executable software component subject to receiving certification within each gaming machine, each software component subject to receiving certification including a unique identifier;

code signing each executable software component subject to receiving certification with its respective separate and unique PKI certificate, each respective PKI certificate being uniquely identified at least by a unique identifier that is uniquely associated with the executable software component such that identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are associated with identical identifiers and are code signed with identical PKI certificates.

Therefore, claim 17 cannot be said to be anticipated by Rabin et al., for the same reasons as advanced above relative to claim 1. Reconsideration and withdrawal of the rejections applied to claim 17 and its dependent claims are, therefore, respectfully requested.

Independent claim 20

Independent claim 20 has been amended as follows:

20. (Currently Amended) A method for a network connected gaming system to enable only authorized software components of constituent computers of the gaming system to execute, comprising the steps of:

configuring a separate software restriction policy for each authorized software component in each of the constituent computers of the gaming system, and

enforcing the software restriction policy for each authorized software component such that the each authorized software component in each of the constituent computers of the gaming system must be authorized to run by its associated separate software restriction policy.

That is, each authorized software component of each of the computers of the gaming systems must be authorized to run by its own SRP (Software Restriction Policy) – there being one such SRP for each of the constituent authorized software components in each of the computers of the gaming system. In support of the rejection of claim 20, the Examiner again

cites the same passages in Rabin et al.; namely, Col. 26, lines 50-60, Col. 27, lines 30-44, Col. 28, lines 5-15, Col. 28, Table 1, line 30 to Col. 30, line 20, and Col. 52line 60 to Col. 53, line 25.

However, none of these passages support the Office contention that Rabin et al. teach “configuring a separate software restriction policy for each authorized software component” of each computer of the gaming system, as recited herein. Rabin et al. teach the usage of supervision policies, to monitor the process by which tags are received. Col. 9, lines 15-28. Rabin et al. also teach a call-up policy to determine the validity of the tag to determine if a given instance of software may be used:

The step of determining if the attempt to use the instance of the software is allowable can also include the steps of determining if a call-up procedure is needed based on a call-up policy and if so performing a call-up procedure to verify the authenticity and to determine the usage supervision policy of the tag associated with the instance of software. Also included are the steps of updating tag information in the user device based upon an outcome of the call-up procedure an examining status information associated with the tag to determine if use of the instance of software associated with the tag is allowed. Col. 11, lines 22-34.

However, note that Rabin et al. teach a single call-up procedure to determine the validity of all tags and to determine whether all software instances may be used. This is the antithesis of the claimed invention, in that claim 20 calls for “configuring a separate software restriction policy for each authorized software component in each of the constituent computers of the gaming system, and enforcing the software restriction policy for each authorized software component such that the each authorized software component in each of the constituent computers of the gaming system must be authorized to run by its associated separate software restriction policy.”

Rabin et al., therefore, do not teach the subject matter of claim 20. In fact, Rabin et al. specifically teach away from the claimed method, in that Rabin et al. advocate a single call-up

procedure for all instances of software, whereas the claimed invention requires a separate software restriction policy for each authorized software component in each of the constituent computers of the gaming system, as claimed.

Therefore, it is not believed that the 35 USC §102(c) rejection applied to claim 20 and to its dependent claims is tenable. Reconsideration and withdrawal of the rejections applied to claim 20 and its dependent claims are, therefore, respectfully requested.

Independent claim 22

As amended, claim 22 recites:

configuring a separate and unique certificate software restriction policy for each authorized executable software component of each of the constituent computers of the gaming system such that the each authorized executable software component in each of the constituent computers of the gaming system must be authorized to run by its associated separate software restriction policy;

configuring a path software restriction policy to prevent unauthorized software components from executing;

configuring a path software restriction policy to prevent non-explicitly authorized software components from executing;

enforcing the certificate software restriction ~~polices~~ policy configured for each of the authorized executable software components of each of the constituent computers of the gaming system, and

enforcing the path software restriction policies.

The comments immediately above relative to claim 20 are equally applicable here and, in the interest of avoiding duplicative arguments, the arguments advanced relative to claim 20 are repeated herein by reference. Reconsideration and withdrawal of the 35 USC §102(c) rejection applied to claim 22 and to its dependent claims are, therefore, respectfully requested.

Independent claim 24

As amended herewith, claim 24 recites:

producing a separate and unique PKI certificate for each of the plurality of executable software component within the gaming system

subject to receive certification, each respective PKI certificate being associated with a unique identifier that is uniquely associated with the software component such that identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are associated with identical identifiers and are code signed with identical PKI certificates;

code signing each software component subject to receive certification with its respective separate and unique PKI certificate;

configuring a certificate software restriction policy for each of the respective separate and unique PKI certificates, and

enforcing the certificate software restriction policy for each of the respective separate and unique PKI certificates.

To anticipate this claim, Rabin et al. must teach that “a separate and unique PKI certificate is produced for each executable software component” ... “such that identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are associated with identical identifiers and are code signed with identical PKI certificates.”

In Rabin et al., no two tags are identical, as each uniquely identifies an instance of the software with which it is associated. This unique tag (again, no two of them are the same in Rabin et al.) is uniquely associated with the instance of the software and is just that – unique. According to Rabin et al., identical executable software components on different machines would be associated with different tags – exactly the opposite of the claimed invention.

In contradistinction, claim 24 calls for identical authorized executable software components on different machines be code signed with identical PKI certificates, which could never happen in Rabin et al.

Claim 24, therefore, distinguishes over the applied reference to Rabin et al. Reconsideration and withdrawal of the rejections applied to claim 24 and its dependent claims are, therefore, respectfully requested.

Independent claim 25

Claim 25, as amended, recites:

for each of the plurality of gaming machines of the network connected gaming system;

code signing each authorized executable software component with a separate PKI certificate that is unique to the authorized software component such that identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are code signed with identical PKI certificates;

packaging the code signed authorized software components into an installation package;

configuring install policies to install each code signed authorized executable software component contained in the installation package;

configuring certificate rule policies to allow execution of the installed code signed authorized executable software component;

configuring enforcement of the policies.

The arguments advanced relative to claim 24 above are equally applicable here. These arguments, therefore, are incorporated herein by reference.

Independent claim 71

Claim 71 recites a step of:

executing the code signed installation package upon every startup of any of the constituent computers of the gaming system or upon a command, wherein execution of any authorized executable file is predicated upon successfully executing the code signed installation package into which the authorized executable file is packaged.

The Patent Office points to Col. 47, line 56 to Col. 48, line 33 and Col. 52, line 60 to Col. 53, line 25 as teaching such a step. These passages are sufficiently short as to be reproduced herein in their entirety:

In other embodiments, there may be call-up policies (CALL-UP_POLICY_SW) associated with individual instances of software 111-114. In this case, step 272 can examine the rules or tests of the call-up policy (CALL-UP_POLICY_SW) associated with the software content SW or the instance of software (INST_SW) 111-114 that was requested access by a user 213 in step 270. In another embodiment, if the user 213 of a user device 104 attempts to use an untagged instance of software, step 272 may mandate that a call-up is needed. In another embodiment, if the user 213 of

a user device 104 uses tagged software for the first time, then step 272 may mandate that a call-up is needed. In another embodiment, the maximum allowed interval between successive call-up procedures is preferably determined by a combination of elapsed time in a user device 104, the number and duration of uses to instances of software 111-114, the number of times the device 104 is powered on, and/or by any other measure that is related to time or use of the device 104.

Call-up processing will be discussed in more detail later. Essentially however, during call-up processing, the supervising program (SP) 209 in a user device 104 securely transfers a copy of the tag table 210 and the fingerprint table 126 to the guardian center 103 (FIG. 2). After verification, the guardian center 103 (FIG. 2) compares each tag TAG_INST_SW_n in the tag table 210 against a list of compromised tags. The guardian center 103 (FIG. 2) can detect tags that are invalid or compromised in some manner.

A usage supervision policy POLICY(TAG_INST_SW) associated with each tag can also be checked at the guardian center 103 (FIG. 2) to ensure that tags 120 (and therefore instances of software associated with the tags) are being used in compliance with the usage supervision policy POLICY(TAG_INST_SW). The policy may be for an entire user device 104-107 or on a per user 213 or per tag 120 basis. Also, for untagged software, the fingerprint table 126 can be compared against a fingerprint data structure (explained later) in the guardian center 103 (FIG. 2) to detect uses of infringing software INF_SW. After analysis of the tag table 210 and fingerprint table 126 are complete, the guardian center 103 (FIG. 2) prepares and sends a continuation message (CM) 212 (FIG. 2) back to the user device 104.

and

A call-up is made in accordance with the CALL-UP_POLICY or CALL-UP_POLICY(TAG_INST_SW) as explained above in response to a user's attempt to use an instance of software 111-114 on a user device 104-107. That is, when the user 213 attempts to use an instance of software 111-114 for which the time allowed before the next call-up according to the CALL-UP_POLICY of the user device 104 or the CALL-UP_POLICY(TAG_INST_SW) of the software (SW) for that instance has expired, the supervising program 209 on that device 104-107 initiates step 370. In another embodiment, the SP 209 executes a call-up procedure at a chosen time before the expiration time, regardless of whether a use of an instance of software 111-114 is requested. The CALL-UP_POLICY can be maintained within the supervising program 209 on the user device 104. In addition, it is possible that a call-up may occur because a portion of the supervising program 209, executing regardless of use requests, determines that it is time to perform a call-up. For example, it may take place as the result of a certain number of BOOTUPS (power-ups) of a user device 104-107 having taken place or the first use of untagged software.

If the call-up to the guardian center 103 (FIG. 2) in step 371 fails, then processing proceeds to step 376 where punitive action may be performed by the supervising program (SP) 209 on the user device 104. In the preferred embodiment, the supervising program (SP) 209 will perform a new call-up, retrying several times before beginning punitive action. In the case that punitive action is necessary in step 376, the punitive action may merely be

to inform the user 213 that the instance of software 111-114 that was requested is temporarily inaccessible due to a communications failure.

As the Examiner will note from re-reading these passages, no teaching or suggestion is made therein regarding any “executing the code signed installation package upon every startup of any of the constituent computers of the gaming system or upon a command, wherein execution of any authorized executable file is predicated upon successfully executing the code signed installation package into which the authorized executable file is packaged”, as required by claim 71. Indeed, no code signed installation packages are taught in Rabin et al., nor is there any teaching or suggestion therein of executing such a code signed installation package upon every startup of any of the constituent computers of the gaming system, as again required by claim 71.

Rabin et al. do teach that the supervisory program may execute a call up procedure to verify the tag of an instance of software (which is not the same as a code signed installation package) after a predetermined number of boot ups:

In another embodiment, the SP 209 executes a call-up procedure at a chosen time before the expiration time, regardless of whether a use of an instance of software 111-114 is requested. The CALL-UP_POLICY can be maintained within the supervising program 209 on the user device 104. In addition, it is possible that a call-up may occur because a portion of the supervising program 209, executing regardless of use requests, determines that it is time to perform a call-up. For example, it may take place as the result of a certain number of BOOTUPS (power-ups) of a user device 104-107 having taken place or the first use of untagged software. (Col. 53, lines 3-15).

However, such does not rise to the level of a teaching of executing a code signed installation package upon every startup of any of the constituent computers of the gaming system, as claimed herein. Recall that the tags in Rabin et al. are associated with instances of software, and not to code signed installation packages. in view of the foregoing, it is respectfully submitted that claim 71 is not anticipated by Rabin et al. Reconsideration and withdrawal of the

35 USC §102(e) rejection of claim 71 over the Rabin et al. reference is, therefore, respectfully requested.

Independent claim 73

Claim 73 recites:

packaging the authorized executable files into a code signed installation package;
configuring certificate rule policies to enable execution of the code signed installation package;
configuring enforcement of the policies, and
re-installing the code signed installation package at every startup of any of the constituent computers of the gaming system or upon a command, wherein execution of any authorized executable file is predicated upon successfully executing the code signed installation package into which the authorized executable file is packaged.

The arguments advanced relative to claim 71 are repeated here, by reference. Claim 73 calls for re-installing a code signed installation package at every startup of the constituent computers of the gaming system. In contrast, Rabin et al. teach that a Supervisory Program (SP 209) executes a call-up after a predetermined number of boot-ups to verify the software instance's tag. Claim 73, however, requires that a code signed installation package (≠ Rabin et al.'s Supervisory Program 209 or tags) be re-installed at every startup of any of the constituent computers. By definition, therefore, Rabin et al. cannot anticipate claim 73. Reconsideration and withdrawal of the 35 USC §102(e) rejection of claim 73 over the Rabin et al. reference is, therefore, respectfully requested.

Independent claim 75

Independent claim 75 includes similar recitations as does claim 73:

packaging the at least one non-executable file into at least one code signed installation package;
configuring certificate rule policies to enable execution of the at least one code signed installation package;

**configuring enforcement of the policies, and
re-installing the at least one code signed installation package at every
startup of any of the constituent computers of the gaming system or upon a
command.**

Rabin et al. demonstrably do not teach re-installing the code signed installation packages(s) upon every startup, as claimed. The arguments advanced immediately above relative to claims 71 and 73 are incorporated herein by reference. Reconsideration and withdrawal of the 35 USC §102(e) rejection of claim 75 over the Rabin et al. reference is, therefore, respectfully requested.

Independent claim 77

Independent claim 77, similarly, recites:

**...
re-installing the at least one code signed installation package at every
startup of any of the constituent computers of the gaming system or upon a
command.**

The arguments advanced above relative to claims 71, 73, and 75 are also applicable here, and are incorporate herein by reference, as if repeated here in full. Reconsideration and withdrawal of the rejections applied to claim 77 and its dependent claims are, therefore, respectfully requested.

Independent claim 79 and its dependent claims

Claim 79 recites:

**packaging at least one authorized non-executable file that control
controls the scheduling of the at least one authorized executable software
component into at least one code signed installation package, each of the at
least one code signed installation packages including a predetermined PKI
certificate;**

**configuring certificate rule policies to enable execution of the at least
one code signed installation package in selected ones of the plurality of
gaming machines; and**

configuring enforcement of the certificate rule policies; and

**downloading the at least one code signed installation package into the selected ones of the plurality of gaming machines;
executing the at least one code signed installation package.**

It is respectfully submitted that Rabin et al. do not teach packaging one or more authorized non-executable files that control the scheduling of the at least one authorized executable software component into at least one code signed installation package, as claimed – and much less that each of the code signed installation packages includes a predetermined PKI certificate, as also claimed.

The Examiner repeatedly (and in support of the rejection of this claim also) cites Col. 26, lines 50-60, Col. 27, lines 30-44, Col. 28, lines 5-15, Col. 28, Table 1, line 30 to Col. 30, line 20, and Col. 52line 60 to Col. 53, line 25 as teaching the claimed subject matter. However, none of these passages teach packaging one or more authorized non-executable files that control the scheduling of the authorized executable software components as required by the claim. This subject matter simply is not present in Rabin et al. If the Office maintains its position that such claimed subject matter is indeed taught by Rabin et al., the Office is respectfully requested to specifically point out where such teachings of packaging one or more authorized non-executable files that control the scheduling of the authorized executable software components are to be found in this reference. Failing a teaching of this recitation, an anticipation rejection cannot stand.

Reconsideration and withdrawal of the anticipatory rejections applied to claim 79 and to its dependent claims are, therefore, respectfully requested.

Independent claim 82 and its dependent claims

Independent claim 82, as amended herewith, recites:

code-signing means for enabling the manufacturer or subcontractor to associate a separate and unique PKI certificate with each authorized software component subject to regulatory certification such identical authorized software components subject to regulatory certification in different ones of the plurality of gaming machines of the network connected gaming system are code signed with identical PKI certificates.

Therefore, to anticipate this claim, Rabin et al. must teach code-signing means ... to associate a separate and unique PKI certificate with each authorized software component *such identical authorized software components subject to regulatory certification in different ones of the plurality of gaming machines of the network connected gaming system are code signed with identical PKI certificates*, as claimed. In Rabin, each instance of software is assigned a unique tag, as detailed in Col. 10, lines 31-67:

Embodiments of the invention also encompass a tag server that accepts a copy of specific vendor software and produces a plurality of tags, one tag per instance of the software, with each tag uniquely identifying an instance of software with which it is associated.

...

In the method, tag creation includes steps of assigning a unique number to the instance of software and computing a first hash function value on portions of the content of the instance of software. Then computing a second hash function value for the instance of software, the second hash function value combining the name of the software, the unique number of the instance of software, and the first hash function value. Next, the method includes the step of computing a tag that is uniquely associated with the instance of software, the tag including the name of the software, the unique number of the instance of software and the second hash value.

That is, in Rabin et al., no two tags are identical, as each uniquely identifies an instance of the software with which it is associated. This unique tag (no two of them are the same in Rabin et al.) is constructed by assigning a unique number, computing a first hash function and a second hash function, after which the tag is computed using the name of the software, the unique number and the second hash value. This tag that is uniquely associated with the instance of the software is just that – unique.

In contradistinction, claim 82 calls for identical authorized software components subject to regulatory certification in different ones of the plurality of gaming machines of the network connected gaming system to be code signed with identical PKI certificates, which could never happen in Rabin et al.

Claim 82, therefore, distinguishes over the applied reference to Rabin et al. Reconsideration and withdrawal of the rejections applied to claim 82 and its dependent claims are, therefore, respectfully requested.

Independent claim 94

Independent claim 94 recites:

generating a unique code signed PKI certificate for a predetermined software module of each authorized game;
generating an executable companion file for each authorized game, wherein the executable companion file is configured to execute faster than the authorized game;
code signing both the predetermined software module and its executable companion file with the generated PKI certificate;
enforcing software restriction policy rules for preventing non-authorized software components from executing;
enforcing software restriction policy rules for enabling execution of selected ones of the authorized games;
attempting to execute each executable companion file, and
adding only those games to the menu of authorized games whose executable companion file has not been denied execution by the software restriction policy rules.

The claim requires that an executable companion file be generated for each authorized game – and that both the predetermined software module and this generated executable companion file be code signed with the generated PKI certificate. Rabin et al. do not teach any such executable companion file, and much less an executable companion file that has been code signed with the same PKI certificate as a software module of the game.

The claim also requires that the executable companion file execute faster than the authorized game, which is something that Rabin et al. do not teach at all. Moreover, Rabin et al. do not teach any step of attempting to execute any companion file and adding only those games to the menu of authorized games whose executable companion file has not been denied execution by software restriction policies.

Note paragraphs [0122] to [0124] in the originally-filed specification for an explanation of the claimed “companion file”:

[0122] Fig. 20 shows a companion Hello component, according to another aspect of the present invention. Reference numeral 2000 in Fig. 20 illustrates a method to generate a code signed companion software component. Each game comprises an aggregate of executable and non-executable software components, usually comprising files such as *.exe, *.dll, *.dat, *.xml. In general, all the software components are dependent of one component named the main program or the game entry. Starting the execution of the main game component is a lengthy process, as a large number of dependent executable components and graphics need to be verified (SRP verification) and started. ...

[0123] Another embodiment of the present invention, therefore, provides a method to quickly verify the policy enforcement on a game without starting the entire game, in order to generate the list of available games to be made available to the player in a menu. For each game, a very short companion .dll file may be created having, for example, only one line of code « Return “HELLO” » which would return the exemplary “HELLO” string when called. Assuming “Infinity.dll” 2010 is the main game component file name 2002 (or friendly name), then the companion file may be named “Infinity.Hello.dll” 2018. ...

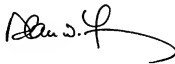
[0124] It is to be noted that code signing two distinct software executables with the same certificate is a deviation from the method taught earlier in this document. However, the fact that the role of the companion file is very well defined, as having for example only one line of code « Return “HELLO” » which would return the “HELLO” string when called, this does not present an issue with the regulators or the certification lab.

Rabin et al. do not teach anything of the sort. Moreover, none of the passages identified by the Examiner even remotely teach any type of code-signed executable companion file, the successful execution of which is a necessary predicate to adding a game to a menu of authorized games, as required by claim 94. It is respectfully submitted that Rabin et al. simply do not teach

any such subject matter. Reconsideration and withdrawal of the rejections applied to claim 94 and its dependent claims are, therefore, respectfully requested.

Applicants' attorney believes that the present application is now in condition for an early allowance and passage to issue. If any unresolved issues remain, the Examiner is respectfully invited to contact the undersigned attorney of record at the telephone number indicated below, and whatever is required will be done at once.

Respectfully submitted,



Date: September 26, 2006

By: _____

Alan W. Young
Attorney for Applicants
Registration No. 37,970

YOUNG LAW FIRM, P.C.
4370 Alpine Rd., Ste. 106
Portola Valley, CA 94028
Tel.: (650) 851-7210
Fax: (650) 851-7232

\\Ylfserver\y\lf\CLIENTS\JMG\CYBS\5858 (Trusted Game Download)\5858 AMEND.2.doc